# Wake on LAN over Internet as Web Service

Juan Antonio Gil-Martínez-Abarca,
Computer Science and Technology Department, University of Alicante
P.O. Box 99, E-03080, Alicante, Spain
gil@dtic.ua.es

Diego Marcos-Jorquera
Computer Science and Technology Department, University of Alicante
P.O. Box 99, E-03080, Alicante, Spain
dmarcos@dtic.ua.es

Francisco Maciá-Pérez
Computer Science and Technology Department, University of Alicante
P.O. Box 99, E-03080, Alicante, Spain
pmacia@dtic.ua.es

Virgilio Gilart-Iglesias
Computer Science and Technology Department, University of Alicante
P.O. Box 99, E-03080, Alicante, Spain
vgilart@dtic.ua.es

## Abstract

*In this paper we present an approach based on the use of embedded network devices for the deployment of small network services, such as DHCP, BOOTP, filters or very specific proxies. The novelty of the proposal resides in the very reduced size of the devices, in that every device is specialized in a certain network function, and specially designed to operate with minimum maintenance, and in the fact that they are presented under both conventional (client-server) and more open (SOA) standards, more concretely, as Web Services. At the same time, these embedded services can work in an individual way or in collaboration with other enterprise network services, either provided by means of conventional systems or by means of other embedded devices. To specify the proposal, we have chosen as example an embedded device able to manage the remote boot of network nodes by means of Wake on LAN (WoL) through Internet.*

## 1. Introduction

The importance of Information and Communication Technologies (ICT) in all areas of human activity in today's world is an indisputable fact. The simpler these technologies are for end users the more complex become the backend systems which support them. These technologies provide the technological base on which most business processes are sustained. It is extremely important to provide the necessary mechanisms to ensure that this infrastructure is in continuous operation.

Although the services provided by the ICT may become extremely complex, they tend to be supported by small, more or less standardised services, which carry out repetitive, quite clearly defined tasks and which usually serve as a support for high level applications —for example, names services, network configuration service, time synchronizing service, activity monitoring service, route calculations or discovery services.

The use of embedded devices providing very specific services is not only a novelty but it is becoming an increasingly accepted alternative for solving specific network problems, without excessively complicating their maintenance. In this way a number of appliances are assuming the critical functions of organisation: intrusion detection systems, filters, communications servers, etc.

The novelty of our proposal resides in the fact that we generalised this approach to all those small network services described above which lack entity for themselves.

In order to ensure the viability of our proposal it was based it on very small embedded network devices (the size of a simple lighter) and low costs. These small network devices are specialised in a single service and are designed to provide minimum intervention possible by those responsible for their management.

Incorporating a service of this kind into our organisation or even at home is simply a question of selecting the appropriate device and connecting it to our network. System administrators need not concern themselves with the hardware or software platform of servers and clients, the initial set up, subsequent maintenance of the service, their integration with other services and, due to the intrinsically robust nature of the service offered as hardware, even managing to avoid the time consuming and laborious maintenance tasks required by physical breakdowns in the device.

In the last ten years there have been tremendous advances in technologies for developing small network

devices, with a more than acceptable capacity for calculation, autonomous operation [1] and the possibility of embedding intelligence in those devices. Although until recently the cost of these devices did not justify their incorporation en masse for handling specific tasks and services, the current trend towards increasingly small devices, with greater computational and communication capabilities and with competitive prices, has led to an ideal scenario for consideration of our proposal as a viable option.

This paper develops this proposal through the design and implementation of a specific network device in which a service has been embedded for remote booting of nodes. This device is known as WoLI —pending patent application (P200501234, 2005)— and provides Wake on LAN (WoL) on Internet, that is, the capacity to handle the remote booting of nodes in a LAN, from any location by means of Internet Standard protocols.

In the following paragraphs we review the current status of the technologies involved, we describe the WoLi service, the structure of the hardware and software device in which it is embedded, the application protocol used, and its conception as Web Services. Finally, we detail the conclusions arising from the research and the lines of work we are currently pursuing.

## 2. Background

The first open standards which attempted to address problems of ICT management in a global manner were SNMP and CMIP [2], proposed by the IETF; both protocols being principally oriented towards network monitoring and control. The main inconvenience of these administration models was their dependence on the platform.

Based on these, and seeking an integration with heterogeneous systems, two principal lines of work arose : an attempt to achieve integration of the systems using the same network management protocol, as is the case of [3] and [4] with the use of CORBA; and more ambitiously, to propose a network management protocol which would be independent of the infrastructures. Some of the more extensive proposals include: CORBA/JIDM, specification of the work group JIDM [5] of the OMG [6]; CIM/WBEM, proposal of the DMTF [7] using techniques oriented towards CIM objects and interoperation using HTTP and XML with WBEM; JMX specification defined by the JCP [8] which defines a series of API's oriented to Java for network management; and WS-Management specification carried out by various companies in the sector (SUN, INTEL, MS, AMD) for the integration of service management systems and resources based on Web Services.

The considerable number of tasks associated with network management, as well as their extremely diverse and complex nature, makes maintenance of these systems expensive for organisations, both in terms of resources and time and personnel. The use of multi-agent systems for computer network management provides a series of characteristics which favour automation and self reliance in maintenance processes [9] [10]. The creation of projects such as AgentLink III, the first Coordinated Action on based on Agents financed by the 6th European Commission Framework Programme [11], is a clear indicator of the considerable degree of interest in research into software agents.

More recently, with the development of Web technologies, further progress has been made towards self-management, proposing self organisation models based on ontologies as information models and on SOA as an operational model, in addition to providing administrators with an ubiquitous interface which does not depend on any platform [12].

The diversity of existing network management models shows that there is patently a need for defining mechanisms which interoperate between all the management domains involved [13]. This relation may reach a semantic level using ontologies in such a way that it is possible to work with an abstract view of the network management information, independent of the specific model used, and which will enable administrators (persons or software) to automate management tasks [14]. Semantic Management permits the integration of common policies for management of network resources and services by rendering the models independent through a common and consensual ontology, [15] which implies comprehension of the network elements and permits normalisation of the tasks to be implemented.

In [16], a proposal is made for a group of basic operations for a Web Service to be standardised within the management networks as a counterpart to standardisation of the SNMP information model under XML development in other works [17].

In other areas of automated information handling in which various devices may intervene, such as the industrial and domotic processes, the development of autonomous management has focused on service-oriented architecture for embedded systems [18], [19], reflecting in the first of these the importance of transferring these ideas to the sphere of autonomous management of networks and systems. It is precisely on these lines that we can frame our proposal: development of network services embedded in hardware, more specifically a remote booting service of nodes which will permit its integration with existing network management models.

## 3. Description of the WoLI Service

The WoLI service is a Web Service which enables boot control of the network nodes with WoL support, using for this purpose Internet standard protocols and service-oriented architectures, which render the service regardless of the administrator's location or the platform used by the administrator.

In addition, this service will be embedded in a specific network device, which is small in size, transparent to existing ICT infrastructures and with minimum maintenance required from the system administrators.

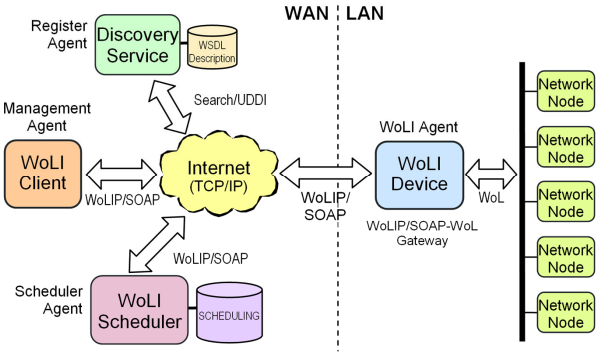


**Figure 1. Organization of the functional elements of the WoLI service.**

The principal usefulness of the WoLI service is to facilitate remote management of network nodes through a wide area network, in general, and Internet in particular, in which the simple impossibility of starting up, disconnecting or restarting nodes notably restricts the system administrators' capacity to act remotely.

Figure 1 shows a diagram of the main elements and actors involved in the service, together with the existing relation between them. We may synthesise these as: WoLI clients, WoLI schedulers, WoLI device and the network nodes for which the service is destined. Below we analyse each of these elements.

The **WoLI client** provides the user, through the service management agent, access to the service, both for schedule, and in order to generate WoL instructions via Internet. Orders are transmitted to the WoLI scheduler or to the WoLI device by means of the WoLIP application protocol (Wake on LAN over Internet Protocol) defined for this purpose, embedded in SOAP messages. This agent will generally be external to the device.

The **WoLI Scheduler** usually acts as a control panel for all the possible WoLI devices distributed through Internet. This control is implemented through the scheduler agent which carries out, executes and verifies all the previously established tasks on the WoLI devices (switches on an individual node or a group of nodes, verifies its status, updates the firmware of the WoLI devices and even plans the work of the WoLI device). This agent may reside in an external node outside the LAN, generally at the location of the service or communications provider, or it may be integrated in a WoLI device. From the point of view of a WoLI client, the scheduler agent behaves like a Web Service which can be located thanks to the UDDI register.

The **WoLI Device** is the cornerstone of the service. This is an embedded network device which is small in size and is designed to act as a WoLIP-WoL pathway between the broad area network and the local area network in which it is located. This agent acts as a Web Service, receiving instructions defined in the WoLIP protocol, encapsulated in SOAP messages and defined by means of WSDL pages.

In the specific case of an instruction to remotely boot a network node, the agent undertakes to translate the request to a WoL datagram.

The **WoLI agent** always acts from within the LAN and behaves differently according to whether it acts in active or passive mode. In passive mode, the agent awaits reception of WoLIP requests from a management agent, generally external, in order to execute them on demand (see figure 2). However, in active mode, it is the agent which takes the initiative and requests a work plan from a scheduler agent. Its operation in active mode is fundamental, as it enables working independently of the possible security policies implemented for protection of the intranet, since for this purpose it would use standard HTTP requests as a base for the WoLIP protocol.

The **network nodes** are the object of the administration and comprise all those devices connected to the network with WoL support in their adapting cards. We are talking about PCs, network servers, networking devices or any other device which fulfils the established requirements.

The **Discovery Service** comprises a standard UDDI registration service. It is responsible for maintaining the pages describing the WoLI services in WSDL format, as well as facilitating that information to the clients wishing to access the service.

## 4. WoLIP Service protocol

The WoLIP service protocol defines a series of instructions used by users and by the various components of the system in order to communicate with each other. This protocol is based on messages and is supported on SOAP, which acts as an information exchange mechanism permitting remote procedure calls.

Each WoLIP protocol command is embedded in the body of a SOAP message which contains the name of a remote procedure, which implements the functionality of the command and the arguments required for its

execution.

*SOAP Envelope*
*SOAP Msg Header*
*SOAPMsg Body*
*WoLIP Commnad*
*Action level 1*
*Action level 2*
*Arguments*

It is possible to distinguish the following elements:

- **Command** defines the service actions in terms of request. Corresponds to the name of the remote procedure which implements the functionality of the WoLIP command.
- **Action level 1** and **Action level 2** are special parameters which profile the functionality of the request.
- **Argument** represents the necessary information for executing the application.

For each WoLIP request message there will be a corresponding SOAP response, the body of which will be formed by the request made or by an error message should there be any problem in execution.

The requests defined in the protocol (see table 1) may be grouped into three major types: configuration orders, basic orders and control orders.

The configuration of the internal variables of the device determines their function mode. These variables are managed using the **SET** command. The basic service provided by the WoLI is invoked using the **WAKE** command. Below we present an example of a SOAP request which invokes the WAKE command.

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
Soap:encodingStyle="http://schemas.xmlsoap.org/soap/">
   <soap:Body>
        <wp:wake xmlns:wp="http://www.dtic.ua.es/wolip">
             <host>192.168.1.23</host>
        </wp:wake>
   </soap:Body>
</soap:Envelope>
```

The **PUT** and **GET** commands, combined with the **SCHDL** argument for programming and obtaining the programming of a device, enable the connection process of one or several sets of equipment to be planned in an autonomous manner.

The service permits access control by means of a user-password type list. These variables are managed using the **VALIDATE** command.

All the WoLIP protocol orders provide a response: OK, if the order is correctly executed or conversely ERROR if it is not, except **GET SCHDL** which returns the list of tasks programmed for the device and the **PING** command which will asynchronously return the status of the network node it has attempted to switch on.

**Table I. Main commands supported by WoLIP Protocol**

| CMD | ARG | FUNCTION |
|---|---|---|
| SET | MODE | Reports the current operation mode. |
| | MODE PASSIVE [puerto] | Sets the passive mode and, optionally, the listening port number. |
| | MODE ACTIVE <ip> [:Puerto] | Sets the active mode, specifying the server's IP address and port number. |
| | RUN | Reports the current WoL service state. |
| | RUN <STARTS \| STOP> | Starts or stops the WoL service. |
| UPDT | FIRM <file> | Updates the device's firmware, from the specified file. |
| GET | SCHDL | Returns the list of scheduled tasks in the device. |
| PUT | SCHDL | Adds a task or a set of tasks to the scheduling. |
| VALIDATE | <user> <pass> | User identification and authentication. |
| WAKE | <host> | Boots a network node through WoL. |
| PING | <host> | Checks if the specified network node is running. |

Figure 2 shows a simple sequence diagram in which a WoLI client or a previously programmed Scheduler requests remote boot of network device through a WoLI device, which translates the request so that its LAN nodes understand it. In addition the WoLI device can check that the node does in fact start up, communicating this to the requester.

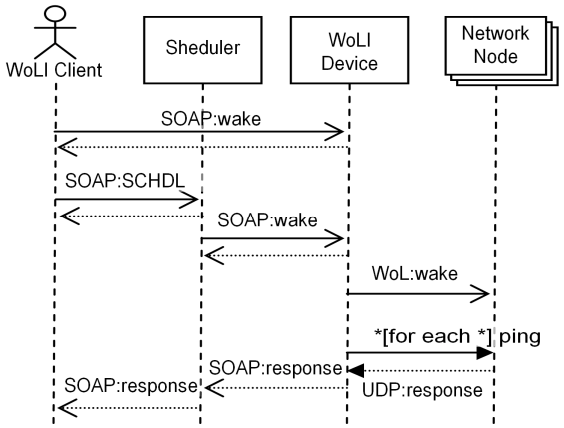


**Figure 2. Sequence Diagram of the network node remote boot, using the WoLI agent's passive mode.**

## 5. WoLI device

In the following paragraphs we shall focus on the design and the implementation of both software and hardware for a totally functional prototype of this embedded network device.

### 5.1. Hardware design

The hardware platform supporting the WoLI device (fig. 3) basically consists of a microcontroller, a volatile memory (SRAM), another permanent memory (ROM) and a rewritable memory (Flash), in addition to a network interface.

The hardware platform chosen for the prototype development is a Lantronix Xport device which has a 16 bit DSTni-EX processor with 48/88MHz frequency reaching 12/22MIPS respectively. The various memory modules provided by this device undertake specific tasks according to their intrinsic features: the execution programmes and the data handled by the device reside in the SRAM memory (256KB); the ROM memory (16KB) contains the system start up programme, and finally, the EEPROM memory with 512KB stores information which though non-volatile, is susceptible to change, such as the set up of the WoLI device or the system programmes which may be updated. These capacities are sufficient for the memory requirements of the software developed, providing approximately 192K of free RAM memory for implementing the protocol.

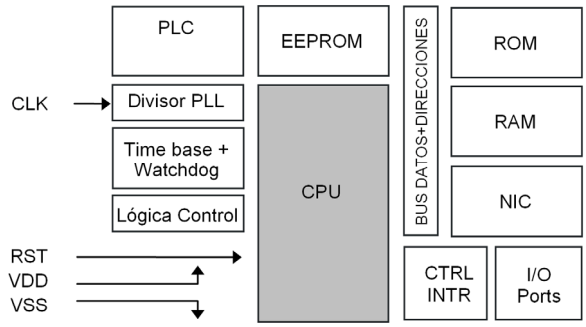


**Figure 3. Hardware architecture of WoLI device.**

As communication elements between subsystems it has a data and address bus which connects the CPU to the RAM, ROM and EEPROM memories; in addition, for CPU external communication, it has various E/S mechanisms, one of which is a FastEthernet network interface which allows more acceptable external communications ratios. In addition, in order to ensure correct operation of the system there is a series of auxiliary elements such as: a watchdog which monitors the CPU and prevents it from blocking; a EEPROM (PLC) manager which controls its updates; a PLL frequency divider required to set up the frequency of the system clock, and which facilitates energy saving; a control unit which supervises the signals required for correct operation of all the elements, and a switch controller which facilitates handling of the entry/exist mechanisms.

Finally, the system has four lines of entry: an adjustable clock signal (CLK) to optimise consumption or performance according to needs; a restart signal (RST) and the power voltages (VDD) and the power mass (VSS).

### 5.2. Software design

The software architecture of this device (fig. 4) consists of various functional elements organised under a hierarchic model.

The system agents are located in the upper layers providing the device services, together with utilities and auxiliary applications which facilitate maintenance. Basically two agents can be distinguished:

- WoLI Agent, which activates the network nodes by generating standard protocol packages for remote boot (WoL).
- Scheduler Agent, which enables planning of the device tasks, specifying times and conditions for execution.

This layer relies mainly on implementation of the WoLIP service protocol defined in the previous section (see table 1). This service protocol is based on messages in XML format encapsulated in SOAP messages.

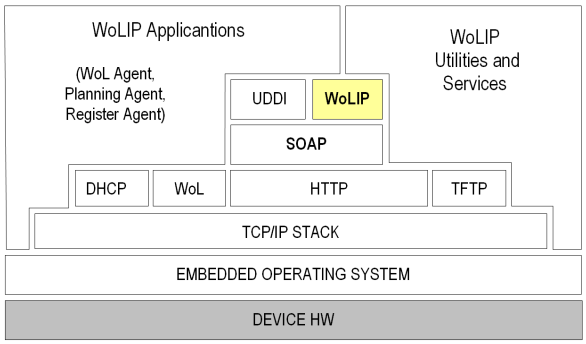


**Figure 4. Architecture software of WoLI device.**

Web Services have been chosen to the detriment of other options such as JINI and uPnP, due to the fact that the former is dependent on the platform for which it is designed: JAVA; and the second comprises various protocols which, in reality are not standard, but consist of their own implementations [20].

Despite the extended capacities provided by the present level of miniaturisation hardware, the devices continue to present some limitations in their resources. For this reason it was decided to implement functionality which is strictly necessary for compliance with the Web Services standard.

Development was based on the cSOAP library suitable for this type of device [21]. However, some changes were made to the library due to the restrictions imposed by the device (limitations on the use of memory, dynamic memory allocate etc.) These restrictions have required us to substitute the XML

parser on which the cSOAP were based, LibXML2 (with a size greater than 1 MB), for another parser with more limited functions but with enough to achieve our objective, C-XML (with 28 Kb). We have also had to adapt both the cSOAP library and the C-XML parser to the proprietary libraries provided by the firmware of the device used.

One of the more notable restrictions is the fact of having to do without the UDP protocol. Since we are unable to count on cSOAP implementation with the UDP protocol, we cannot use discovery mechanisms (ws-discovery), events (ws-eventing) and set up (ws-addresing) specific to the standard Web Service, and therefore a minimal but compatible solution has been proposed, based on DHCP and AutoIP as network configuration protocols and UDDI for registering and advertising the services.

For this reason, the embedded device only implements the UDDI functionality, which allows to automatically publish the WSDL document describing the service provided by the WoLI device, without requiring the participation of an external entity. The service definition was carried out using WSDL and registered with UDDI, however, the embedded device does not have the WSDL analyser, nor does it consult the UDDI registry due to the fact that, in principle, it will not locate and invocate other Web services. Below we show how the WSDL page is defined for the **WAKE** operation of the WoLIP protocol:

```
<operation name="wake">
        <SOAP:operation style="rpc" soapAction=""/>
        <input>
                <SOAP:body use="encoded"
                namespace="urn:woli" \
                encodingStyle="http://schemas.xmlsoap.org/so
                ap/encoding/"/>
        </input>
        <output>
                <SOAP:body use="encoded"
                namespace="urn:woli" \
                encodingStyle="http://schemas.xmlsoap.org/so
                ap/encoding/"/>
        </output>
</operation>
```

In order to accomplish its task; the WoLI agent will make use of the implementation of the standard protocol for remote boot of the network nodes (WoL). In this way, it acts as a path between the standard protocols and the WoL method.

In addition, the device has been provided with various services based on widely used protocols which support utilities and management tools. The aim is for the device to support conventional management mechanisms, so that they may be easily integrated in the management policies currently in use. According to this, in order to store or recover the device configuration or to update its firmware, the light file transfer protocol has been used (TFTP); and for the device management the light network management protocol has been provided (SNMP), which will facilitate the management and administration tasks of the network device.

In the lower layers more general protocols are defined such as the TCP/IP stack, the embedded operating system of the device –which in the case in question is CoBOS, owned by Lantronix, and which already includes an implementation for the TCP/IP stack, with DHCP, AutoIP and BOOTP as protocols for automatic network configuration.

Finally, we have the device hardware analysed in the previous section, together with the primitives which facilitate access to its resources.

## 6. WoLI plug and play

Since the embedded devices improve and provide sufficient resources, it is important for the WoLI service to be implemented with all the functionality of standard Web Services. In fact, our final goal is to employ Semantic Web Service techniques, together with an ontology which defines the processes and the network administration information, to provide completely autonomous services, able to register on command like utility computing, to discover other services and to interact with them, all simply by connecting the embedded device to the communications network, without the need for human intervention at any time.

Web services of the type described in the present work do present some shortcomings in resepect of reliability, interoperability, security orchestration and semantics. In order to improve these parameters, and specifically the autonomy and standardisation of the device so that Wolf becomes a self configuring device able to collaborate with other devices and processes in a highly distributed and autonomous environment, it is necessary to incorporate into the device the extensions [20] required to address and transport messages, discovery, events, description and control.

There is no standard definition of the Web Services Protocol stack though the W3C Web Services Architecture Working Group did publish a Web Services Architecture document which provides an excellent context for the various protocols. Other proposals exist, as those of [16] [18] [20], with similar architecture. These should not be seen as proprietary, as the underlying standards are common. Our own interpretation is shown in Figure 5.

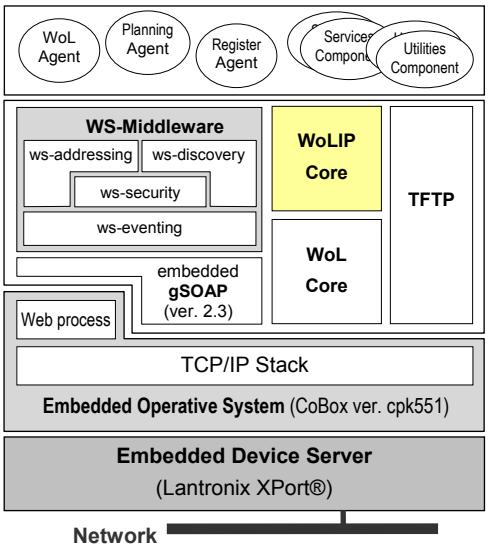


**Figure 5. Software architecture of the production devices.**

All that is required for the WoLI device is WS-Addressing for the automatic configuration of devices and message transport; WS-Eventing to enable management of events which are interesting for the system such as for example, informing of the finalisation of start up of the equipment; WS-Discovery which enables discovery and use of the device by simply connecting the device to the network and finally, in order to to improve security and ensure that this is achieved end to end, the WS-Security extension should be incorporated.

In order to implement those extensions one of the main changes introduced was the use of the cSOAP programming library by gSOAP.

The gSOAP compiler optimises the code generated in order to reduce occupied memory. gSOAP 2.3 includes WSDL1.1, SOAP1.1/1.2, a Web server with HTTP 1.0 and 1.1 and an XML parser, as well as functionality for handling plug-ins which enable incorporation of WS-Addressing and WS-Security. gSOAP permits programming of the remaining extensions based on UDP as WS-Eventing, since, contrary to what occurs with cSOAP, it provides SOAP implementation over UDP, which is essential for this type of extension [22].

## 7. Conclusions

In this paper we have presented an application for the provision of simple network services (DHCP, NTP, Jini LUS, etc.) in embedded form in small size, energy saving network devices which are autonomous and specialised for each service.

The main objective of this application is that it can provide these services, which are common to any current network area (manufacturing industrial networks, home networks, enterprise networks, etc), without specialist help, either for setting it up or for its subsequent maintenance.

From the user's perspective, each service represents a small device which simply requires connection to the network of their organisation or home to make the required service available.

The device is able to configure itself and work independently, or integrated with other services from existing networks, regardless of whether or not these are provided with other similar devices, or as conventional network services.

In order to verify the validity of this application, the design and implementation of a specific service was discussed, aimed particularly at remote start up of network equipment through Internet (WoL over Internet). This prototype known as WoLI, has been presented in SOA generic form as a Web Service, compatible with open standards such as SOAP, UDDI and WSDL and which has the capacity for self registration.

We are currently working with other embedded network services and integrating them all in a model based on Semantic Web services, so that in future they will not only be compatible with existing services, but also with new services or setups which were not considered in its initial design.

### References


[1] L.B. Ruiz, T.R.M. Braga, F.A. Sikva, H.P. Assuncao, J.M.S. Nogueira, and A.A.F. Loureiro, "On the Design os a Self-Managed Wireless Sensor Network," *IEEE Communications Magazine*, July 2005, pp 95-102.

[2] A. N. Expert, *A Book He Wrote*, His Publisher, 1989.

[3] Jeong, M.S., Kim, K.H., Kwon, J.H., Park, J.T. "CORBS/CMIP: Gateway Service Scheme for CORBA/TMN Integration." *Knom Review*, Vol.2, No. 1, pp. 55-62, 1999

[4] Aschemann, G., Mohr, T., Ruppert, M. "Integration of SNMP into a CORBA- and Web-Based Management Environment," *in Proc. Kommunikation in Verteilten Systemen*, Heidelberg, 1999, pp. 210-221.

[5] Work Group JIDM [Online]. Available: http://www.opengroup.org

[6] OMG [Online]. Available: http://www.omg.org

[7] DMTF [Online]. Available: http://www.dmtf.org

[8] JCP [Online]. Available: http://www.jcp.org

[9] T.C. Du, E.Y. Li, and A.P. Chang, "Mobile Agents in Distributed Network Management", *in Communications at the ACM*, vol. 46, no. 7, 2003, pp. 127-132.

[10] J. Guo,Y. Liao, and B. Parviz. "An Agent-Based Network management system," *in Proceedings of the 2005 Internet and Multimedia Applications*.

[11] European co-ordination action for agent-based computing [Online]. Available: http://www.rfc.net

[12] R. Boutaba, J. Xiao, "Network Management: State of the Art," *in Proceedings of the 2002 World Computer Congress*, pp. 127-146

[13] J.E. López. V.A. Villagrá, and J.I. Asensio, "Ontologies: Giving Semantics to Network Management Models," *IEEE Network*, vol. 17, no. 3, May-June 2003. pp. 15-21.

[14] J. Peer, "A POP-Based Replanning Agent for Automatic Web Service Composition," *in proc. of the 2005 Second European Semantic Web Confenrece*.

[15] A. Guerrero, V.A. Villagrá, and J.E. López, "Definición del comportamiento de gestión de red con reglas SWRL en un marco de gestión basado en ontologías en OWL", *in proc. of the 2005 V Jornadas de Ingeniería Telemática*, Vigo, pp. 677-684.

[16] J. Sloten, A. Pras, and M. Van Sinderen, "On the standardisation of web service management operations," *in proc. of the 2004 X EUNICE Summer School and IFIP WG 6.3 Workshop*.

[17] T. Klie, and F. Straub, "Integrating SNMP agents with XML-based management systems," *IEEE Communications Magazine* vol. 42 Issue 7, July 2004, pp. 76-83.

[18] U. Toop, P. Muller, J. Konnertz, A. Pick, "Web based Service for Embedded Devices," *LNCS* vol. 2593, 2002, pp. 141-153.

[19] F. Jammes et al, "Orchestration of Service-Oriented Manufacturing Process". *In Proc. of the 10th IEEE International Conference on Emerging Technologies and Factory Automation ETFA 2005*, Catania, September 2005, pp. 19-22.

[20] F. Jammes and H. Smit, "Service-Oriented Architectures for Devices- the SIRENA View".

[21] V. Miori, L. Tarrini, and R. Bianchi, "LIGHT: XML-Innovative Generation for home Networking Technologies", *Ercim News*, no. 62, July 2005.

[22] gSOAP [Online]. Available: http://www.cs.fsu.edu/~engelen/soap.html